|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Лабораторная работа №5*

*По предмету: «Анализ алгоритмов»*

**Тема: Конвейерные вычисления**

Студент: Гасанзаде М.А.,

Группа: ИУ7-56Б

Москва, 2019 г.

**Оглавление**

**Введение3**

**1. Аналитическая часть4**

1.1 Описание метода4

1.2 Применение метода4

**2. Конструкторская часть**5

2.1 Разработка алгоритма5

2.3 Вывод6

**3. Технологическая часть**7

3.1 Требования к программному обеспечению7

3.2 Средства реализации7

3.3 Листинг кода7

3.4 Описание тестирования11

3.5 Вывод11

**4. Экспериментальная часть12**

4.1 Примеры работы12

4.2 Результаты тестирования13

4.3 Постановка эксперимента по замеру времени13

4.4 Сравнительный анализ на материале экспериментальных данных14

4.5 Вывод14

**Заключение15**

**Список литературы16**

**Введение**

Параллельные вычисления используют для увеличения скорости исполнения программ. Ведь пока нет возможности сделать один очень быстрый процессор, который можно было бы сравнить с современными параллельными компьютерами. Конвейерная обработка данных является популярным приемом при работе с параллельными машинами.

**Цель работы:** изучение метода конвейерных вычислений.

**Задачи работы:**

1) изучение основ конвейерной обработки данных;

2) получение практических навыков конвейерных вычислений;

3) сравнительный анализ конвейерной и традиционной обработки данных по затрачиваемым ресурсам;

4) экспериментальное подтверждение различий во временной эффективности реализаций вычислений при помощи разработанного программного обеспечения на материале замеров процессорного времени выполнения;

6) описание и обоснование полученных результатов в отчете о выполненной лабораторной работе, выполненного как расчётно-пояснительная записка к работе.

1. **Аналитическая часть**

В данной части дано теоретическое описание метода и указание области его применения.

**1.1. Описание метода**

Конвейеризация – это техника, в результате которой задача разбивается на некоторое число подзадач, которые выполняются последовательно [1]. Каждая подзадача выполняется на своем логическом устройстве. Все логические устройства (ступени) соединяются последовательно таким образом, что выход i-ой ступени связан с входом (i+1)-ой ступени, все ступени работают одновременно. Множество ступеней называется конвейером.

Выигрыш во времени достигается при выполнении нескольких задач за счет параллельной работы ступеней, вовлекая на каждом такте новую задачу. Но в бесконвейерном процессе скорость обработки задач стабильна. Производительность конвейерном подходе предсказать намного сложнее, и она может значительно различаться при разных данных.

**1.2. Применение метода**

Методы активно применяются в [2]:

1. многих задачах линейной алгебры и задачах решения систем уравнений в частных производных;
2. обработке машинных команд процессором.
3. **Конструкторская часть**

Вдохновившись курсом ОС, я решил перенести передачу по передачу сообщений между процессами по трубе, на многопоточную реализацию.

Моделируется передача данных между потоками, полезная работа не производится.

**2.1 Разработка алгоритма**

Функциональная модель процесса передачи сообщения в нотации IDEF0 представлена на рис. 1.

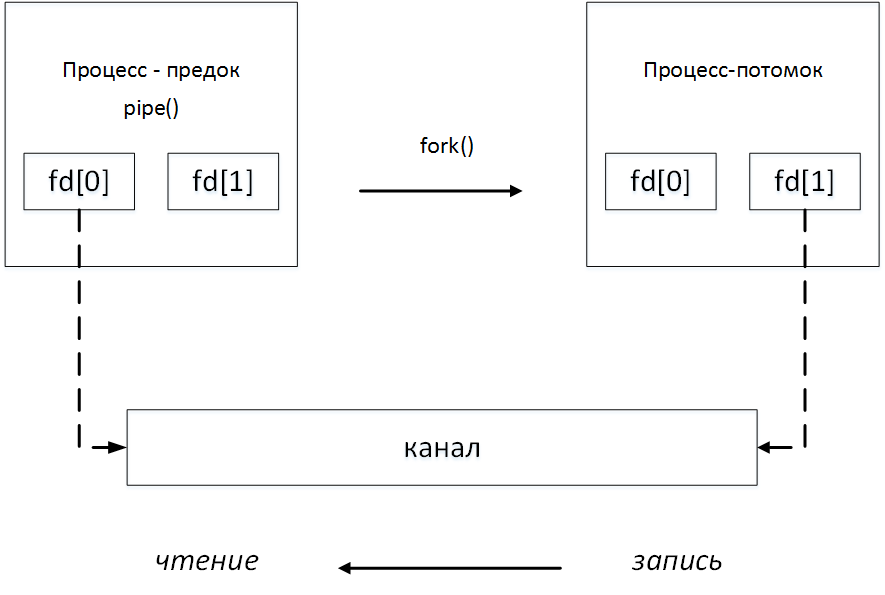
Уровень a, b (посылающие)

Уровень с (получатель)

Pipe(труба)  
()

*Рисунок 1 –* *Функциональная модель процесса передачи сообщений*

Схема алгоритма представлена на рис. 2.



*Рисунок 2 – Схема алгоритма сообщения через трубу*

* 1. **Вывод**

Не смотря на сложность реализации многопоточной реализации, она даст выигрыш, за счёт параллельной обработки ресурсов.

1. **Технологическая часть**

В данной части приведены используемые технические средства, а также примеры тестирования и листинг программы.

**3.1 Требования к программному обеспечению**

Программа должна корректно осуществлять передачу/приём между процессами. Требуется также обеспечить возможность замера времени работы алгоритма.

* 1. **Средства реализации**

Выбран язык программирования Python3 за высокую скорость выполнения программ и широкий выбор инструментов для параллельных вычислений. Работа с потоками осуществляется посредством класса asyncio. Замер времени проводится функцией time.

* 1. ***Листинг кода***

Исходный код реализации программы приведен в листингах 1, 2.

***Листинг 1 – Создание loggerов, загрузка необходимых библиотек***

**import** asyncio

**import** logging

**import** time

**from** concurrent**.**futures **import** ThreadPoolExecutor

logging**.**basicConfig**(**format**=**"[%(thread)-5d]%(asctime)s: %(message)s"**)**

logger **=** logging**.**getLogger**(**'async'**)**

logger**.**setLevel**(**logging**.**INFO**)**

executor **=** ThreadPoolExecutor**(**max\_workers**=**13**)** # thread pool

loop **=** asyncio**.**get\_event\_loop**()** # event loop

***Листинг 2 – Создание трубы и связи между процессами и создание процесса***

**def** cpu\_bound\_op**(**exec\_time**,** **\***data**):**

logger**.**info**(**"Running cpu-bound op on {} for {} seconds"**.**format**(**data**,** exec\_time**))**

time**.**sleep**(**exec\_time**)**

**return** sum**(**data**)**

**async** **def** process\_pipeline**(**data**):**

# just pass the data along to level\_a and return the results

results **=** **await** level\_a**(**data**)** # Waiting for the level a

**return** results

**async** **def** level\_a**(**data**):**

level\_b\_inputs **=** data**,** 2 **\*** data**,** 4 **\*** data

results **=** **await** asyncio**.**gather**(\*[**level\_b**(**val**)** **for** val **in** level\_b\_inputs**])** # aggregate results from the level b

result **=** **await** loop**.**run\_in\_executor**(**executor**,** cpu\_bound\_op**,** 3**,** **\***results**)**

**return** result

**3.4 Описание тестирования:** Тестирование проводится по методу чёрного ящика [3]. Требуется проверить корректность работы на случаях с различной степенью загруженности трубы.

* 1. **Вывод**

Текущая реализация на языке Python3 позволяет корректно передавать сообщения, а также производить замеры времени.

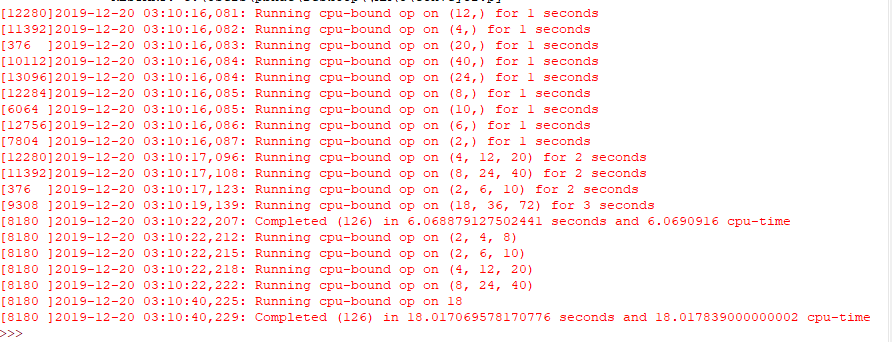
1. **Экспериментальная часть**

В этой части приведены примеры интерфейса, входные данные тестирования, результаты замера времени и их анализ.

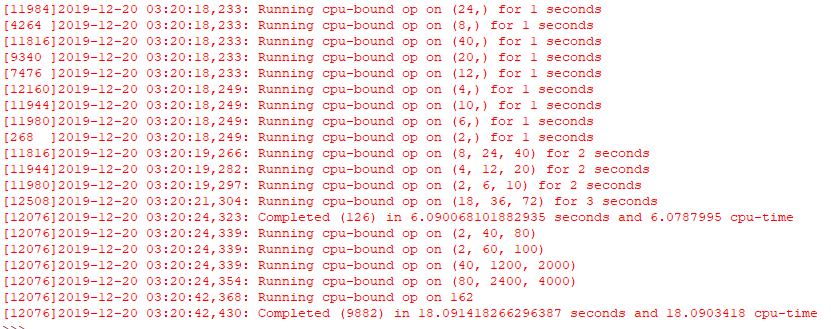
**4.1. Примеры работы**

На рис. 3, 4 приведены изображения внешнего вида интерфейса программы во время его работы.

*Рисунок 3 – Пример работы программы при значении 2&4 и 3&5 данных(элементов в процессе)*



*Рисунок 4 –Пример работы программы при значении 20&40 и 30&50 данных(элементов в процессе)*



*4.2. Результаты тестирования*

В Таблице 1 представлены результаты тестирования.

**Таблица 1.**

**Результаты тестирования по методу черного ящика**

|  |  |  |
| --- | --- | --- |
| Информативная нагрузка проц. | Традиционная реализация | Конвейерная реализация |
| 2&4 3&5 | Передача успешна | Передача успешна |
| 20&40 30&50 | Передача успешна | Передача успешна |

Тесты пройдены успешно.

*4.3. Постановка эксперимента по замеру времени*

*З*амер времени проводился 5 раз, результат одного эксперимента рассчитывался как среднее значение результатов проведенных испытаний с одинаковыми входными данными. Замерялись традиционная реализация и многопоточная реализация.

*4.4. Сравнительный анализ на материале экспериментальных данных*

На рис. 5 представлен результат замеров времени.

Где min: 1&1 1&1

Medium: 2&4 3&5

Medium2: 20&40 30&50

High: 50&50 50&50

*Рисунок 5 – Время передачи сообщений традиционной и конвейерной реализациями*

*4.5 Вывод*

По результатам эксперимента подтверждено, что время исполнения конвейерной реализации значительно меньше времени исполнения традиционной реализации. Для High загруженности контейнерная реализация примерно в 3.29 раза быстрее.

**Заключение**

В рамках данной работы успешно изучены основы контейнерных вычислений. Применен метод динамического программирования для контейнерной реализации для передачи сообщений между процессами. Проведен сравнительный анализ контейнерной и традиционной реализаций. Подтверждены экспериментально различия во временной эффективности реализаций при помощи разработанного программного обеспечения на материале замеров времени выполнения в зависимости от загруженности процесса. Дано описание и обоснование полученных результатов.

Экспериментально получено, что реализация с контейнерами работает более чем три раза быстрее, нежели традиционная.

**Список литературы**

1. Конвейерные вычисления // STUDYLIB URL: https://studylib.ru/doc/4736512/konvejernye-vychisleniya (дата обращения: 19.12.2019).
2. Конвейеризация вычислений // studref URL: https://studref.com/313869/informatika/konveyerizatsiya\_vychisleniy (дата обращения: 19.12.2019).
3. Kara kutu test technologisi URL:<https://www.mobilhanem.com/kara-kutu-test-teknigi-ve-uygulanmasi/> (дата обращения 19.12.2019)